## Linked Lists Adding and Removing Nodes

2 min

With linked lists, because nodes are linked to from only one other node, you can't just go adding and removing nodes willy-nilly without doing a bit of maintenance.

*Adding a new node*
Adding a new node to the beginning of the list requires you to link your new node to the current head node. This way, you maintain your connection with the following nodes in the list.

*Removing a node*
If you accidentally remove the single link to a node, that node's data and any following nodes could be lost to your application, leaving you with orphaned nodes.

To properly maintain the list when removing a node from the middle of a linked list, you need to be sure to adjust the link on the previous node so that it points to the following node.

Depending on the language, nodes which are not referenced are removed automatically. "Removing" a node is equivalent to removing all references to the node.

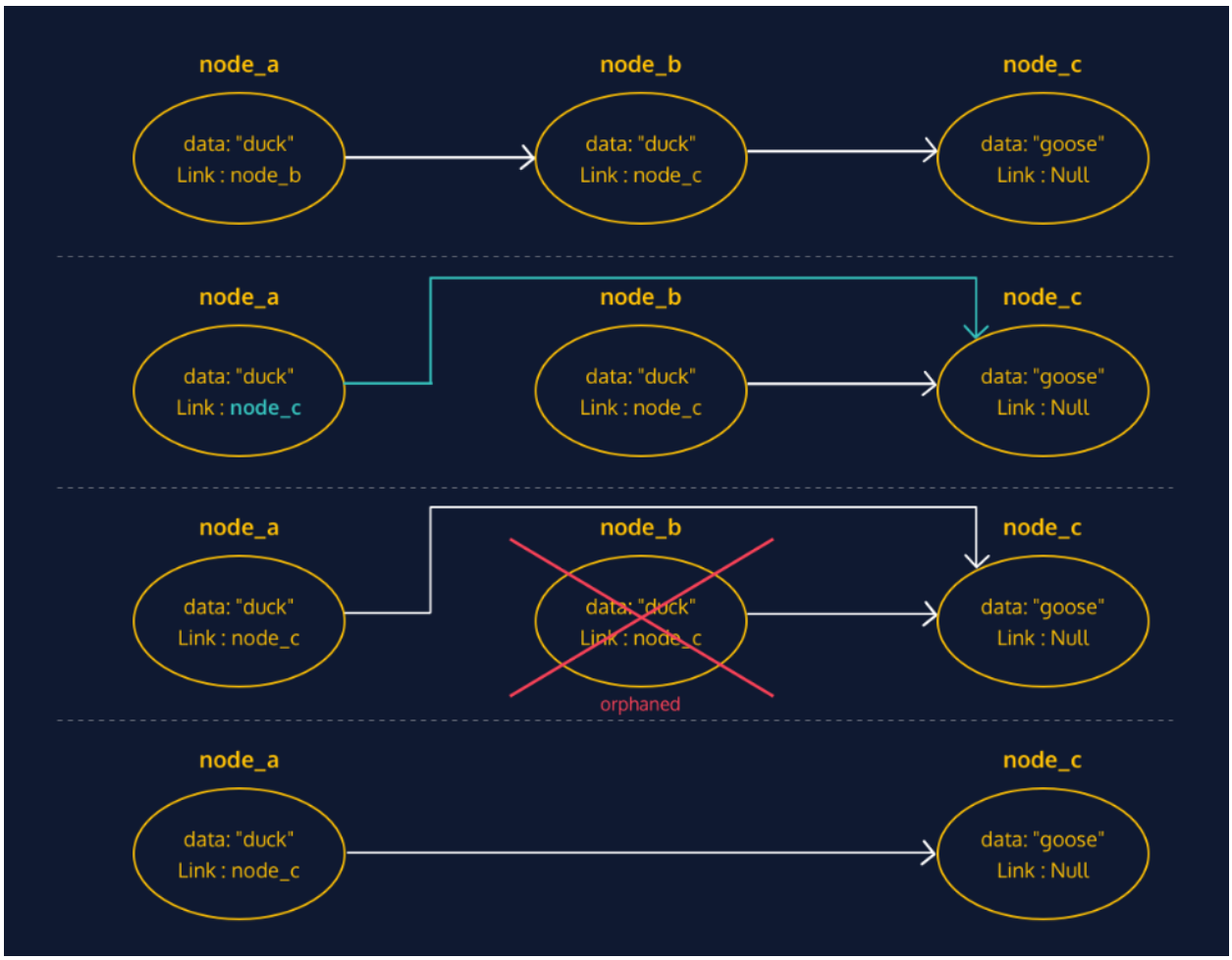
### Instructions

Look at the image to see the proper manner of removing a node.

In order to remove `node_b`, you must first link `node_a` to `node_c` (where `node_b` was linking).

Then you can remove `node_b`.

How would you represent the process of adding a new node to the beginning of a linked list?


node_a
data: "duck"
Link : node_b
node_b
data: "duck"
Link : node_c
node_c
data: "goose"
Link : Null
node_a
data: "duck"
Link : node_c
node_b
data: "duck"
Link : node_c
node_c
data: "goose"
Link : Null
node_a
data: "duck"
Link : node_c
node_b
data: "duck"
Link : node_c
orphaned
node_c
data: "goose"
Link : Null
node_a
data: "duck"
Link : node_c
node_c
data: "goose"
Link : Null